Levels of Software Testing

There are different levels during the process of Testing. In this chapter a brief description is provided about these levels.

Levels of testing include the different methodologies that can be used while conducting Software Testing. Following are the main levels of Software Testing:

- Functional Testing.
- Non-Functional Testing.

Functional Testing

This is a type of black box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved when testing an application for functionality.

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of Test Scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

Unit Testing

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

LIMITATIONS OF UNIT TESTING

Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that the developer can use to verify the source code. So after he has exhausted all options there is no choice but to stop unit testing and merge the code segment with other units.

Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top Down Integration testing.

S.N.	Integration Testing Method
1	**Bottom-up integration** This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
2	**Top-Down integration** This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems and network.

System Testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

System testing is so important because of the following reasons:

- System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

Regression Testing

Whenever a change in a software application is made it is quite possible that other areas within the application have been affected by this change. To verify that a fixed bug hasn't resulted in another functionality or business rule violation is Regression testing. The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.

Regression testing is so important because of the following reasons:

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the change made did not affect any other area of the application.

- Mitigates Risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

Acceptance Testing

This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client.s requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashers or major errors in the application.

By performing acceptance tests on an application the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

ALPHA TESTING

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

BETA TESTING

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.

Non-Functional Testing

This section is based upon the testing of the application from its non-functional attributes. Non-functional testing of Software involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc.

Some of the important and commonly used non-functional testing types are mentioned as follows:

Performance Testing

It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes which contribute in lowering the performance of software:

- Network delay.
- Client side processing.
- Database transaction processing.
- Load balancing between servers.
- Data rendering.

Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

It can be either qualitative or quantitative testing activity and can be divided into different sub types such as **Load testing** and **Stress testing**.

LOAD TESTING

A process of testing the behavior of the Software by applying maximum load in terms of Software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of Software and its behavior at peak time.

Most of the time, Load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the Load testing for the Software. The quantity of users can be increased or decreased concurrently or incrementally based upon the requirements.

STRESS TESTING

This testing type includes the testing of Software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is Stress testing.

The main intent is to test the Software by applying the load to the system and taking over the resources used by the Software to identify the breaking point. This testing can be performed by testing different scenarios such as:

- Shutdown or restart of Network ports randomly.

- Turning the database on or off.
- Running different processes that consume resources such as CPU, Memory, server etc.

Usability Testing

This section includes different concepts and definitions of Usability testing from Software point of view. It is a black box technique and is used to identify any error(s) and improvements in the Software by observing the users through their usage and operation.

According to Nielsen, Usability can be defined in terms of five factors i.e. Efficiency of use, Learn-ability, Memor-ability, Errors/safety, satisfaction. According to him the usability of the product will be good and the system is usable if it possesses the above factors.

Nigel Bevan and Macleod considered that Usability is the quality requirement which can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

Molich in 2000 stated that user friendly system should fulfill the following five goals i.e. Easy to Learn, Easy to Remember, Efficient to Use, Satisfactory to Use and Easy to Understand.

In addition to different definitions of usability, there are some standards and quality models and methods which define the usability in the form of attributes and sub attributes such as ISO-9126, ISO-9241-11, ISO-13407 and IEEE std.610.12 etc.

UI VS USABILITY TESTING

UI testing involves the testing of Graphical User Interface of the Software. This testing ensures that the GUI should be according to requirements in terms of color, alignment, size and other properties.

On the other hand Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of Usability testing.

Security Testing

Security testing involves the testing of Software in order to identify any flaws ad gaps from security and vulnerability point of view. Following are the main aspects which Security testing should ensure:

- Confidentiality.
- Integrity.
- Authentication.
- Availability.
- Authorization.
- Non-repudiation.
- Software is secure against known and unknown vulnerabilities.
- Software data is secure.
- Software is according to all security regulations.

- Input checking and validation.
- SQL insertion attacks.
- Injection flaws.
- Session management issues.
- Cross-site scripting attacks.
- Buffer overflows vulnerabilities.
- Directory traversal attacks.

Portability Testing

Portability testing includes the testing of Software with intend that it should be re-useable and can be moved from another Software as well. Following are the strategies that can be used for Portability testing.

- Transferred installed Software from one computer to another.
- Building executable (.exe) to run the Software on different platforms.

Portability testing can be considered as one of the sub parts of System testing, as this testing type includes the overall testing of Software with respect to its usage over different environments. Computer Hardware, Operating Systems and Browsers are the major focus of Portability testing. Following are some pre-conditions for Portability testing:

- Software should be designed and coded, keeping in mind Portability Requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

Source:

http://www.tutorialspoint.com/software_testing/levels_of_testing.htm